Chapter 3. The File System

Introduction

In this chapter we will look at the file system of UNIX. We also look at types of files their significance. We then look at two ways of specifying a file viz., with absolute pathnames and relative pathnames. A discussion on commands used with directory files viz., cd, pwd, mkdir, rmdir and ls will be made. Finally we look at some of the important directories contained under UNIX file system.

Objectives

- Types of files
- UNIX Filenames
- Directories and Files
- Absolute and Relative Pathnames
- pwd – print working directory
- cd – change directory
- mkdir – make a directory
- rmdir – remove directory
- The PATH environmental variable
- ls – list directory contents
- The UNIX File System

1. Types of files

A simple description of the UNIX system is this:
"On a UNIX system, everything is a file; if something is not a file, it is a process."
A UNIX system makes no difference between a file and a directory, since a directory is just a file containing names of other files. Programs, services, texts, images, and so forth, are all files. Input and output devices, and generally all devices, are considered to be files, according to the system.

Most files are just files, called *regular* files; they contain normal data, for example text files, executable files or programs, input for or output from a program and so on.
While it is reasonably safe to suppose that everything you encounter on a UNIX system is a file, there are some exceptions.
Directories: files that are lists of other files.
Special files or Device Files: All devices and peripherals are represented by files. To read or write a device, you have to perform these operations on its associated file. Most special files are in /dev.
Links: a system to make a file or directory visible in multiple parts of the system's file tree.
(Domain) sockets: a special file type, similar to TCP/IP sockets, providing inter−process networking protected by the file system's access control.
Named pipes: act more or less like sockets and form a way for processes to communicate with each other, without using network socket semantics.

Ordinary (Regular) File

This is the most common file type. An ordinary file can be either a text file or a binary file.

A text file contains only printable characters and you can view and edit them. All C and Java program sources, shell scripts are text files. Every line of a text file is terminated with the *newline* character.

A binary file, on the other hand, contains both printable and nonprintable characters that cover the entire ASCII range. The object code and executables that you produce by compiling C programs are binary files. Sound and video files are also binary files.

Directory File

A directory contains no data, but keeps details of the files and subdirectories that it contains. A directory file contains one entry for every file and subdirectory that it houses. Each entry has two components namely, the filename and a unique identification number of the file or directory (called the *inode number).*

When you create or remove a file, the kernel automatically updates its corresponding directory by adding or removing the entry (filename and inode number) associated with the file.

Device File

All the operations on the devices are performed by reading or writing the file representing the device. It is advantageous to treat devices as files as some of the commands used to access an ordinary file can be used with device files as well.

Device filenames are found in a single directory structure, /dev. A device file is not really a stream of characters. It is the attributes of the file that entirely govern the operation of the device. The kernel identifies a device from its attributes and uses them to operate the device.

2. Filenames in UNIX

On a UNIX system, a filename can consist of up to 255 characters. Files may or may not have extensions and can consist of practically any ASCII character except the / and the Null character. You are permitted to use control characters or other nonprintable characters in a filename. However, you should avoid using these characters while naming a file. It is recommended that only the following characters be used in filenames:

Alphabets and numerals.

The period (.), hyphen (-) and underscore (_).

UNIX imposes no restrictions on the extension. In all cases, it is the application that imposes that restriction. Eg. A C Compiler expects C program filenames to end with .c, Oracle requires SQL scripts to have .sql extension.

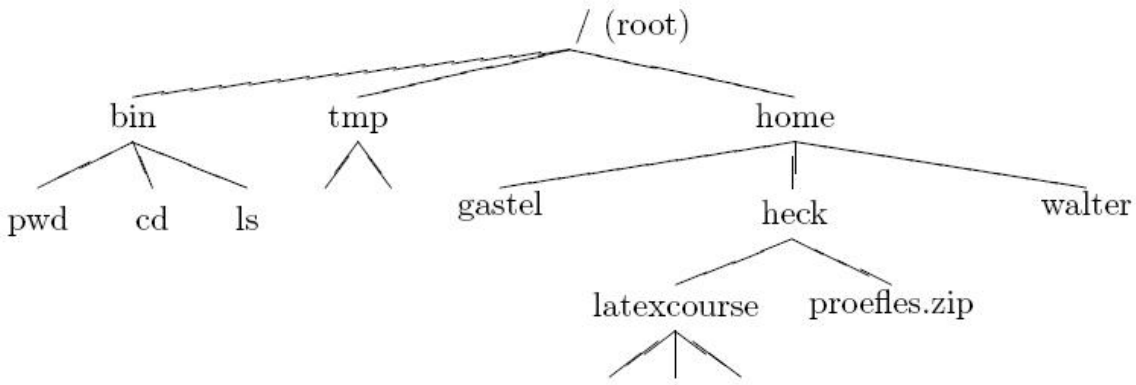
A file can have as many dots embedded in its name. A filename can also begin with or end with a dot.

UNIX is case sensitive; cap01, Chap01 and CHAP01 are three different filenames that can coexist in the same directory.

3. Directories and Files

A file is a set of data that has a name. The information can be an ordinary text, a user-written computer program, results of a computation, a picture, and so on. The file name may consist of ordinary characters, digits and special tokens like the underscore, except the forward slash (/). It is permitted to use special tokens like the ampersand (&) or spaces in a filename.

Unix organizes files in a tree-like hierarchical structure, with the *root directory*, indicated by a forward slash (/), at the top of the tree. See the Figure below, in which part of the hierarchy of files and directories on the computer is shown.



4. Absolute and relative paths

A path, which is the way you need to follow in the tree structure to reach a given file, can be described as starting from the trunk of the tree (the / or root directory). In that case, the path starts with a slash and is called an absolute path, since there can be no mistake: only one file on the system can comply.

Paths that don't start with a slash are always relative to the current directory. In relative paths we also use the . and .. indications for the current and the parent directory.

The HOME variable

When you log onto the system, UNIX automatically places you in a directory called the *home directory*. The shell variable HOME indicates the home directory of the user.

E.g.,

```
$ echo $HOME
  /home/kumar
```

What you see above is an absolute pathname, which is a sequence of directory names starting from root (/). The subsequent slashes are used to separate the directories.

5. pwd - print working directory

At any time you can determine where you are in the file system hierarchy with the ***pwd***, print working directory, command,

E.g.,:

```
$ pwd
  /home/frank/src
```

6. cd - change directory

You can change to a new directory with the ***cd***, change directory, command. ***cd*** will accept both absolute and relative path names.

Syntax

cd [directory]

Examples

cd changes to user's home directory
cd / changes directory to the system's root
cd .. goes up one directory level
cd ../.. goes up two directory levels
cd /full/path/name/from/root changes directory to absolute path named (note the leading slash)
cd path/from/current/location changes directory to path relative to current location (no leading slash)

7. mkdir - make a directory

You extend your home hierarchy by making sub-directories underneath it. This is done with the ***mkdir***, make directory, command. Again, you specify either the full or relative path of the directory.

Examples

mkdir patch Creates a directory *patch* under current directory
mkdir patch dbs doc Creates three directories under current directory
mkdir pis pis/progs pis/data Creates a directory tree with *pis* as a directory under the current directory and *progs* and *data* as subdirectories under *pis*

Note the order of specifying arguments in example 3. The parent directory should be specified first, followed by the subdirectories to be created under it.

The system may refuse to create a directory due to the following reasons:
1. The directory already exists.
2. There may be an ordinary file by the same name in the current directory.
3. The permissions set for the current directory don't permit the creation of files and directories by the user.

8. rmdir - remove directory

A directory needs to be empty before you can remove it. If it's not, you need to remove the files first. Also, you can't remove a directory if it is your present working directory; you must first change out of that directory. You cannot remove a subdirectory unless you are placed in a directory which is hierarchically *above* the one you have chosen to remove.

E.g.

rmdir patch Directory must be empty
rmdir pis pis/progs pis/data Shows error as *pis* is not empty. However rmdir

silently deletes the lower level subdirectories *progs* and *data*.

9. The PATH environment variable

Environmental variables are used to provide information to the programs you use. We have already seen one such variable called HOME.
A command runs in UNIX by executing a disk file. When you specify a command like *date*, the system will locate the associated file from a list of directories specified in the PATH variable and then executes it. The PATH variable normally includes the current directory also.

Whenever you enter any UNIX command, you are actually specifying the name of an executable file located somewhere on the system. The system goes through the following steps in order to determine which program to execute:
1. Built in commands (such as cd and history) are executed within the shell.
2. If an absolute path name (such as /bin/ls) or a relative path name (such as ./myprog), the system executes the program from the specified directory.
3. Otherwise the PATH variable is used.

10. ls - list directory contents

The command to list your directories and files is ***ls***. With options it can provide information about the size, type of file, permissions, dates of file creation, change and access.

Syntax

ls [options] [argument]

Common Options

When no argument is used, the listing will be of the current directory. There are many very useful options for the ls command. A listing of many of them follows. When using the command, string the desired options together preceded by "**-**".

-a Lists all files, including those beginning with a dot (**.**).
-d Lists only names of directories, not the files in the directory
-F Indicates type of entry with a trailing symbol: executables with *, directories with / and symbolic links with @
-R Recursive list
-u Sorts filenames by last access time
-t Sorts filenames by last modification time
-i Displays inode number
-l Long listing: lists the mode, link information, owner, size, last modification (time). If the file is a symbolic link, an arrow (-->) precedes the pathname of the linked-to file.

The **mode field** is given by the **-l** option and consists of 10 characters. The first character is one of the following:

CHARACTER	**IF ENTRY IS A**
d	directory
-	plain file
b	block-type special file

c character-type special file
l symbolic link
s socket

The next 9 characters are in 3 sets of 3 characters each. They indicate the **file access permissions**: the first 3 characters refer to the permissions for the **user**, the next three for the users in the Unix **group** assigned to the file, and the last 3 to the permissions for **other** users on the system.

Designations are as follows:

r read permission
w write permission
x execute permission
- no permission

Examples

1. To list the files in a directory:

```
$ ls
```

2. To list all files in a directory, including the hidden (dot) files:

```
$ ls -a
```

3. To get a long listing:

```
$ ls -al
total 24
drwxr-sr-x 5 workshop acs 512 Jun 7 11:12 .
drwxr-xr-x 6 root sys 512 May 29 09:59 ..
-rwxr-xr-x 1 workshop acs 532 May 20 15:31 .cshrc
-rw------- 1 workshop acs 525 May 20 21:29 .emacs
-rw------- 1 workshop acs 622 May 24 12:13 .history
-rwxr-xr-x 1 workshop acs 238 May 14 09:44 .login
-rw-r--r-- 1 workshop acs 273 May 22 23:53 .plan
-rwxr-xr-x 1 workshop acs 413 May 14 09:36 .profile
-rw------- 1 workshop acs 49 May 20 20:23 .rhosts
drwx------ 3 workshop acs 512 May 24 11:18 demofiles
drwx------ 2 workshop acs 512 May 21 10:48 frank
drwx------ 3 workshop acs 512 May 24 10:59 linda
```

11. The UNIX File System

The root directory has many subdirectories. The following table describes some of the subdirectories contained under root.

Directory	Content
/bin	Common programs, shared by the system, the system administrator and the users.
/dev	Contains references to all the CPU peripheral hardware, which are represented as files with special properties.
/etc	Most important system configuration files are in /etc, this directory contains data similar to those in the Control Panel in Windows
/home	Home directories of the common users.
/lib	Library files, includes files for all kinds of programs needed by the system and the users.
/sbin	Programs for use by the system and the system administrator.

/tmp	Temporary space for use by the system, cleaned upon reboot, so don't use this for saving any work!
/usr	Programs, libraries, documentation etc. for all user-related programs.
/var	Storage for all variable files and temporary files created by users, such as log files, the mail queue, the print spooler area, space for temporary storage of files downloaded from the Internet, or to keep an image of a CD before burning it.

Conclusion

In this chapter we looked at the UNIX file system and different types of files UNIX understands. We also discussed different commands that are specific to directory files viz., pwd, mkdir, cd, rmdir and ls. These commands have no relevance to ordinary or device files. We also saw filenaming conventions in UNIX. Difference between the absolute and relative pathnames was highlighted next. Finally we described some of the important subdirectories contained under root (/).